Set builder notation provides instructions on how to build a set. As such, it should not be part of our Set class. Our class should give the user the ability to do some basic set operations:

1. Conjunction
2. Inclusive disjunction
3. Exclusive disjunction
4. Determine if this set is a subset of another.
5. Determine if this set if a proper subset of another.

(More Later)

Def of subset: . That is, A is a subset of B if every element in A is also an element in B. If a is a subset of B, then A cannot have elements that aren’t in B. It’s denoted . This allows for A and B to be equal.

Def of proper subset: . That is, A is a proper subset of B, denoted , if A every element in A is also in B and A and B are not equal (|A| < |B|).

Getting the cardinality of our set is to return the size of the underlying set.

denote the powerset of the set S. The powerset of S is the set of all possible subsets of S, including S itself and the null set. The cardinality of the powerset is .

The empty set, , has one subset, namely itself P() = {}.

P({}) = {}. The powerset of the set containing the empty set contains the empty set and the set containing the empty set. Replace the set containing the empty set with the set containing any other element.

Sets are unordered. Ordered n-tuples are ordered. n is the length of the tuple. Performing a cartesian product between two sets, or a set and itself, will produce a set of ordered pairs (the pairs are 2-tuples). This is where you pair each element in the left operand with each element in the right operand. The cartesian product of sets A and B is denoted A x B. Unless A = B,

A x B B x A. The cartesian product is the set of all ordered pairs from A and B such that

Let’s say we allow our Set class to produce a cartesian product. We can use arrays of size 2 to try and represent this, or we can create an inner OrderedPair class to represent it. The latter option would lend itself easier to generics.